Lecture 12

Reinforcement Learning-Part 1

Reinforcement Learning

- Reinforcement Learning from Human Feedback (RLHF).

Deep Reinforcement Learning

- Deep RL is a combination of RL and DL

- **TD-Gammon** is a game learning program consisting of a neural network that is able to teach itself to play backgammon solely by playing against itself and learning from the results.

- **Deep Q-Network (DQN)** is the first deep reinforcement learning method proposed by DeepMind and used in Atari games.

Alpha Go

- **Alpha Go** is a computer system developed by Google DeepMind that can play the game Go.
  - Google DeepMind's Challenge Match, was a five-game Go match between 18-time world champion Lee Sedol and AlphaGo played in 2016.
  - AlphaGo won all but the fourth game.

Alpha Go Zero

- **Alpha Go. Zero** is more powerful and is arguably the strongest Go player in history.
    - Alpha Zero learns to play simply by playing against itself, starting completely from random play.

Reinforcement Learning Definition

Wikipedia:

Reinforcement learning is an area of machine learning inspired by behavioural psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

Reinforcements

- We use reinforcements to train animals
  - Food: (Positive reinforcements)
  - Hunger (Negative reinforcements)

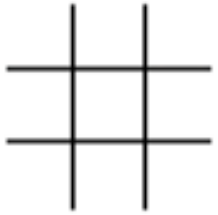
Tic-tac-toe

Tic-tac-toe

- Environment
- Action
- State
- Reward

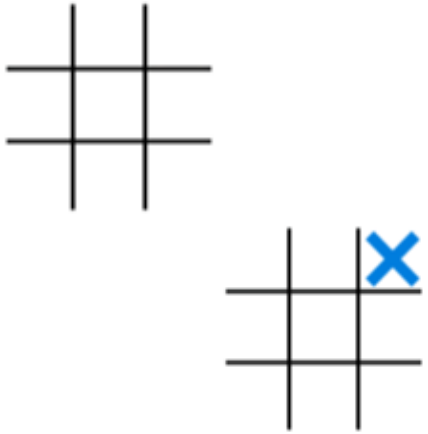
Tic-tac-toe

- Environment



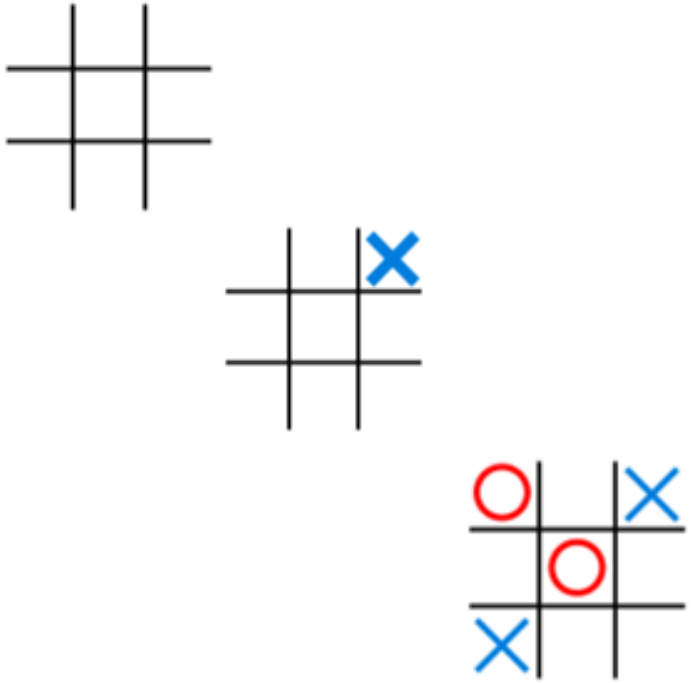
Tic-tac-toe

- Environment
- Action



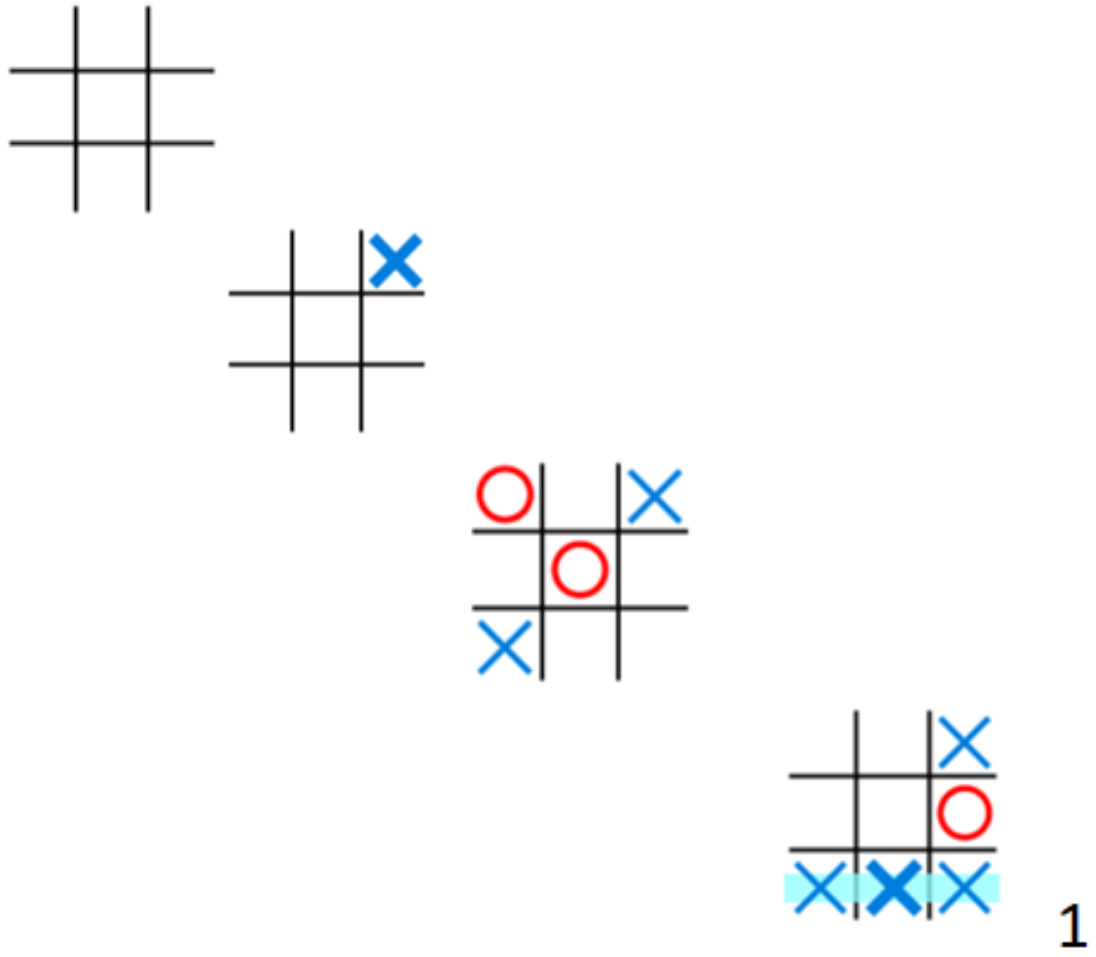
Tic-tac-toe

- Environment
- Action
- State



Tic-tac-toe

- Environment
- Action
- State
- Reward



1

Markov Decision Process (MDP)

MDP is defined by a quintuple $\{S, A, R, P, \gamma\}$.

- States: $s \in S$
- Actions: $\mathrm{a} \in A$
- Rewards: $\mathrm{r} \in R$
  - Reward model: $\Pr(r_t | s_t, a_t)$
- Transition model: $\Pr(s_t | s_{t-1}, a_{t-1})$
- Discount factor: $0 \leq \gamma \leq 1$
  - Discounted: $\gamma < 1$ Undiscounted: $\gamma = 1$

Markov Decision Process (MDP)

- States: $s \in S$
  - The set of all possible situations (s) an agent can encounter.
- Actions: $\mathrm{a} \in A$
- Rewards: $\mathrm{r} \in R$
  - Reward model: $\Pr(r_t|s_t, a_t)$
- Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
- Discount factor: $0 \leq \gamma \leq 1$
  - Discounted: $\gamma < 1$ Undiscounted: $\gamma = 1$

Markov Decision Process (MDP)

- States: $s \in S$
  - The set of all possible situations (s) an agent can encounter.
- Actions: $\mathrm{a} \in A$
  - The set of all possible moves (a) the agent can make.
- Rewards: $\mathrm{r} \in R$
  - Reward model: $\Pr(r_t|s_t, a_t)$
- Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
- Discount factor: $0 \leq \gamma \leq 1$
  - Discounted: $\gamma < 1$ Undiscounted: $\gamma = 1$

Markov Decision Process (MDP)

- States: $s \in S$
  - The set of all possible situations (s) an agent can encounter.
- Actions: $\mathrm{a} \in A$
  - The set of all possible moves (a) the agent can make.
- Rewards: $\mathrm{r} \in R$
  - Reward model: $\Pr(r_t|s_t, a_t)$
    - Scalar feedback (r) received after executing an action in a state
- Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$

- Discount factor: $0 \leq \gamma \leq 1$
  - Discounted: $\gamma < 1$ Undiscounted: $\gamma = 1$

Markov Decision Process (MDP)

- States: $s \in S$
  - The set of all possible situations (s) an agent can encounter.
- Actions: $\mathrm{a} \in A$
  - The set of all possible moves (a) the agent can make.
- Rewards: $\mathrm{r} \in R$
  - Reward model: $\Pr(r_t|s_t, a_t)$
    - Scalar feedback (r) received after executing an action in a state
- Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
  - The probability of moving to the next state (s_t) from the current state-action
- Discount factor: $0 \leq \gamma \leq 1$
  - Discounted: $\gamma < 1$ Undiscounted: $\gamma = 1$

Markov Decision Process (MDP)

- States: $s \in S$
  - The set of all possible situations (s) an agent can encounter.
- Actions: $\mathrm{a} \in A$
  - The set of all possible moves (a) the agent can make.
- Rewards: $\mathrm{r} \in R$
  - Reward model: $\Pr(r_t|s_t, a_t)$
    - Scalar feedback (r) received after executing an action in a state
- Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
  - The probability of moving to the next state (s_t) from the current state-action
- Discount factor: $0 \leq \gamma \leq 1$
  - A coefficient ($0 \leq \gamma \leq 1$) determining the present value of future rewards.
  - Discounted: $\gamma < 1$ Undiscounted: $\gamma = 1$

Markov Decision Process (MDP)

- At every time step $t = 0, 1, \ldots,$ the agent is at state S_t.

Markov Decision Process (MDP)

- At every time step $t = 0, 1, \dots,$ the agent is at state S_t.

- Takes an action A_t

Markov Decision Process (MDP)

- At every time step $t = 0, 1, \dots,$ the agent is at state S_t.
- Takes an action A_t
- Transitions to a new state S_{t+1}, following the probability $S_{t+1} \sim \mathcal{P}(\cdot \,|S_t, A_t)$

Markov Decision Process (MDP)

- At every time step $t = 0, 1, \dots,$ the agent is at state S_t.
- Takes an action A_t
- Transitions to a new state S_{t+1}, following the probability $S_{t+1} \sim \mathcal{P}(\cdot \,|S_t, A_t)$
- Obtains a reward $R_t \sim \mathcal{R}(\cdot \,|S_t, A_t)$

Policy

- A policy π is a mapping from states to actions,
- $A_t = \pi(S_t)$ (deterministic policy)
- $A_t = \pi(\cdot\,|S_t)$ (stochastic policy).

Markov Decision Process (MDP)

- **Mathematical Formulation of the Objective:**
  - Find a policy π such that the long-term reward of the agent is maximized.
  - $\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t=0}^{h} \gamma^t E_\pi[r_t]$

- **Policy:**
  - A policy π dictates the agent's action in each state.
  - The objective is to determine an optimal policy π^* that maximizes expected long-term rewards.

Examples of Deterministic and Stochastic Policies

Deterministic Policy:

Scenario: Robot in a maze.

Policy: At a junction, turn right. At a dead-end, turn around.

Characteristic: No randomness, fixed actions.

Stochastic Policy:

Scenario: Marketing strategy for customer interactions (state).

Policy: For a sports product browser, 70% chance to suggest related items, 20% for fitness services, 10% for supplements.

Characteristic: Actions based on probabilities, varied responses.

Bellman's Equation

Definition:

The Bellman equation is a fundamental recursive formula in reinforcement learning that calculates the optimal value of a current state by considering all possible future states. It balances the immediate reward with the maximum expected future rewards.

$$V(s_t) = \max_{a_t} \left[R(s_t, a_t) + \gamma \sum_{s_{t+1}} Pr(s_{t+1}|s_t, a_t) V(s_{t+1}) \right]$$

Bellman’s Equation in Action

1. **Morning Choices:**
    1. Visit the museum (low cost, informative).
    2. Go to the amusement park (high cost, fun-filled).
2. **Considering the Afternoon:**
    1. If you choose the museum, you'll have more time and money left for other activities.
    2. The amusement park is more expensive and time-consuming, leaving less flexibility for the rest of the day.
3. **Decision-Making:**
    - Balancing immediate rewards with future possibilities

Bellman’s Equation in Action

1.Quantify Outcomes:

1. Assign values to the outcomes (e.g., enjoyment, cost, time).
2. **Museum: 40 points** (savings on time and money, educational value).
3. **Amusement Park: 50 points** (thrills and fun, higher cost, less time for other activities).

2.Consider Future Rewards:

1. Estimate the remaining day's potential with either choice.
2. Museum: Extra time and money could lead to 30 more points (e.g., visiting a park, enjoying a nice meal).
3. Amusement Park: Fewer resources might limit you to 10 more points (e.g., a quick street food dinner).

3.Calculate Total Value:

1. Museum: 40 (immediate) + 0.9 x 30 (future) = 67 points.
2. Amusement Park: 50 (immediate) + 0.9 x 10 (future) = 59 points.

Bellman's Equation

$$V(s_t) = \max_{a_t} \left[R(s_t, a_t) + \gamma \sum_{s_{t+1}} Pr(s_{t+1}|s_t, a_t) V(s_{t+1}) \right]$$

Value Iteration

$\mathbf{V}^*(s) \leftarrow 0$ **for all** s

repeat until convergence

$$\mathbf{V}^*(s) \leftarrow \max_a \left[R(s,a) + \gamma \sum_{s'} \Pr(s' \mid s,a)\mathbf{V}^*(s')\right]$$

return $\mathbf{V}^*$

$\pi^*(s) = \arg\max_a [R(s,a) + \gamma \sum_{s'} Pr(s'|s,a)V^*(s')]$ for all s.

Note: Optimal policy π^* changes with time. It's non-stationary.

State-Value VS Optimal State-Value

State-Value Function $V^{\pi}(s)$

- Represents the expected total reward for a state s under policy π.

Optimal State-Value Function $V^{*}(s)$

- The highest expected reward attainable from state s under any policy.

Policy Iteration

1. **Policy Evaluation:**
   - Update value function based on current policy.
   - $$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^{\pi}(s') \quad \forall s$$
2. **Policy Improvement:**
   - Adjust policy based on updated value function.
   - $$\pi(s) \leftarrow \arg\max_{a} \left[R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^{\pi}(s') \right] \quad \forall s$$

Modified Policy Iteration

1. **Partial Policy Evaluation:**
   - Repeat k times:

$$V^{\pi}(s) \leftarrow R(s, \pi(s)) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) V^{\pi}(s') \quad \forall s$$

2. **Policy Improvement:**
   - Update the policy based on evaluated values:

$$\pi(s) \leftarrow \arg\max_{a} \left[R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi}(s') \right] \quad \forall s$$

Markov Decision Process

- Definition
  - States: $s \in S$
  - Actions: $\mathrm{a} \in A$
  - Rewards: $\mathrm{r} \in R$
  - Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
  - Reward model: $\Pr(r_t|s_t, a_t)$
  - Discount factor: $0 \leq \gamma \leq 1$

  - Goal: find optimal policy π^* such that

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t=0}^{h} \gamma^t E_\pi[r_t]$$

Reinforcement Learning

- Definition
  - States: $s \in S$
  - Actions: $\mathrm{a} \in A$
  - Rewards: $\mathrm{r} \in R$
  - Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
  - Reward model: $\Pr(r_t|s_t, a_t)$
  - Discount factor: $0 \leq \gamma \leq 1$

  - Goal: find optimal policy π^* such that

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t=0}^{h} \gamma^t E_\pi[r_t]$$

Model Free

- Given a policy π, how can we estimate $V^{\pi}(s)$ without transition model?

Monte Carlo Estimation for State-Value Function

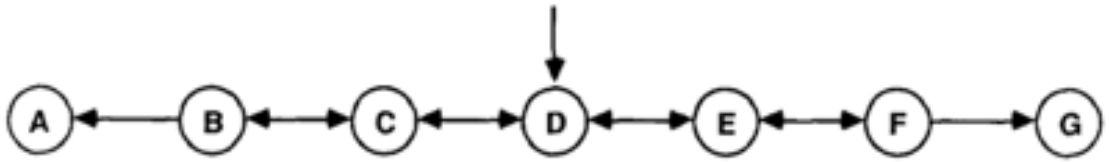
$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t} \gamma^{t} r_{t}\right]$$

$$V^{\pi}(s) \approx \frac{1}{n(s)} \sum_{k=1}^{n(s)} \left[\sum_{t} \gamma^{t} r_{t}^{(k)}\right]$$

- Computes the average return for state s based on sampled episodes.
- $n(s)$: Number of episodes where state s was visited.
- $r_t^{(k)}$: Reward at time t in episode k.

Monte Carlo Estimation for State-Value Function

$$V^{\pi}(s) \approx \frac{1}{n(s)} \sum_{k=1}^{n(s)} \left[\sum_{t} \gamma^{t} r_{t}^{(k)} \right] = \frac{1}{n(s)} \sum_{k=1}^{n(s)} [G_k]$$



Monte Carlo Estimation for State-Value Function

- Let G_k be a one-trajectory Monte Carlo target

$$G_k = \sum_t \gamma^t r_t^{(k)}$$

- Approximate value function

$$\begin{aligned} V_n^\pi(s) &\approx \frac{1}{n(s)} \sum_{k=1}^{n(s)} G_k \\ &= \frac{1}{n(s)} \left(G_{n(s)} + \sum_{k=1}^{n(s)-1} G_k \right) \\ &= \frac{1}{n(s)} \Big(G_{n(s)} + (n(s)-1) V_{n-1}^\pi(s) \Big) \\ &= V_{n-1}^\pi(s) + \frac{1}{n(s)} \Big(G_{n(s)} - V_{n-1}^\pi(s) \Big) \end{aligned}$$

- **Incremental update**

$$V_n^\pi(s) \leftarrow V_{n-1}^\pi(s) + \alpha_n (G_n - V_{n-1}^\pi(s))$$

$1/n(s)$

Incremental update

$$V_n^\pi(s) \leftarrow V_{n-1}^\pi(s) + \alpha_n(G_n - V_{n-1}^\pi(s))$$

Temporal Difference (TD) evaluation

$$V_n^\pi(s) \leftarrow V_{n-1}^\pi(s) + \alpha_n(r + \gamma V_{n-1}^\pi(s') - V_{n-1}^\pi(s))$$

Incremental update

$$V_n^\pi(s) \leftarrow V_{n-1}^\pi(s) + \alpha_n(G_n - V_{n-1}^\pi(s))$$

Temporal Difference (TD) evaluation

$$V_n^\pi(s) \leftarrow V_{n-1}^\pi(s) + \alpha_n(r + \gamma V_{n-1}^\pi(s') - V_{n-1}^\pi(s))$$

$$V^\pi(s) = E[r|s, \pi(s)] + \gamma \sum_{s'} \Pr(s'|s, \pi(s)) V^\pi(s')$$
$$\approx r + \gamma V^\pi(s')$$

Temporal Difference Evaluation

- **Incremental update**

$$V_n^\pi(s) \leftarrow V_{n-1}^\pi(s) + \alpha_n(r + \gamma V_{n-1}^\pi(s') - V_{n-1}^\pi(s))$$

- **Theorem:** if α_n is appropriately decreased with the number of times a state is visited, then $V_n^\pi(s)$ converges to correct value
- **Sufficient conditions** for α_n:

$$(1)\ \sum_n \alpha_n \to \infty \qquad (2)\ \sum_n (\alpha_n)^2 < \infty$$

- Often $\alpha_n(s) = 1/n(s)$
  - Where $n(s) =$ # of times s is visited

Intuitively

- instead of trying to calculate total future reward, TD simply tries to predict the combination of immediate reward and its own reward prediction at the next moment in time.

- when the next moment comes, the new prediction is compared against what it was expected to be. (temporal difference)

- use this “temporal difference” to adjust the old prediction toward the new prediction.

Temporal Difference Evaluation

Tdevaluation(π, V^{π})

- Repeat
  - Execute $\pi(s)$
  - Observe s' and s
  - Update counts: $n(s) \leftarrow n(s) + 1$
  - Learning rate: $\alpha \leftarrow 1/n(s)$
  - Update value: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha\big(r + \gamma V^{\pi}(s') - V^{\pi}(s)\big)$

    $s \leftarrow s'$
- Until convergence of V^{π}
- Return V^{π}

Comparison: V^π and Q^π

- **State-Value Function** $V^\pi(s)$:
  - Represents the expected return starting from state s and following policy π thereafter.
  - Defined as: $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| S_t = s \right]$
  - Depends only on the current state.
- **Action-Value Function** $Q^\pi(s, a)$:
  - Represents the expected return after taking action a in state s and following policy π thereafter.
  - Defined as: $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| S_t = s, A_t = a \right]$
  - Depends on both the current state and the action taken.

State-action value $Q^{\pi}(s,a)$

- Instead of evaluating the state value, $V^{\pi}(s)$, evaluate the state-action value, $Q^{\pi}(s,a)$

$Q^{\pi}(s,a)$: the value of executing a followed by π

$$Q^{\pi}(s,a) = E[r|s,a] + \gamma \sum_{s'} \Pr(s'|s,a)\, V^{\pi}(s')$$

- Optimal policy π':

$$\pi'(s) = \operatorname*{argmax}_{a} Q^{\pi}(s,a)$$

Bellman's Equation

- Optimal state value function $V^*(s)$

$$V^*(s) = E[r|s,a] + \gamma \sum_{s'} \Pr(s'|s,a) V^*(s')$$

- Optimal state-action value function $Q^*(s,a)$

$$Q^*(s,a) = E[r|s,a] + \gamma \sum_{s'} \Pr(s'|s,a) \max_{a'} Q^*(s',a')$$

where $V^*(s) = \max_a Q \quad (s,a)$

$$\pi^*(s) = \operatorname{argmax}_a Q \quad (s,a)$$

Monte Carlo

- Let G_n^a be a one-trajectory Monte Carlo target

$$G_n^a = \underbrace{r_0^{(n)}}_{a} + \underbrace{\sum_{t=1} \gamma^t r_t^{(n)}}_{\pi}$$

- Alternate between
  - **Policy Evaluation**

$$Q_n^\pi(s,a) \leftarrow Q_{n-1}^\pi(s,a) + \alpha_n\left(G_n^a - Q_{n-1}^\pi(s,a)\right)$$

  - **Policy Improvement**

$$\pi'(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q^\pi(s,a)$$

Temporal Difference

- Approximate Q-function:

$$Q\ \ (s,a) = E\ [r|s,a] + \gamma \sum_{s'} \Pr(s'|s,a) \max_{a'} Q\ \ (s',a')$$

$$\approx r + \gamma\ \max_{a'} Q\ \ (s',a')$$

- **TD**

$$Q_n\ (s,a) \leftarrow Q_{n-1}(s,a) +\ \alpha_n \left(r + \gamma \max_{a'}(s',a') - Q_{n-1}(s,a) \right)$$

Q-Learning

1. Initialize Q values arbitrarily for all state-action pairs.
2. Repeat until convergence or for a sufficient number of episodes:
   - Choose an action a from state s
   - Take action a, observe reward r and next state s'.
   - Update Q values using the rule:

     $$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

   - Update the state: $s \leftarrow s'$
3. Continue with the new state.

Exploration vs. Exploitation

Exploration:

- **Definition:** Taking random actions to discover new paths and outcomes.
- **Advantage:** Uncovers new possibilities and avoids local optima.
- **Disadvantage:** May lead to immediate suboptimal rewards.

Exploitation:

- **Definition:** Choosing actions that promise the highest reward based on current knowledge.
- **Advantage:** Maximizes short-term rewards and makes use of known information.
- **Disadvantage:** Risks missing out on potentially better options.